Using uFR library in Xcode

1.0

Digital Logic Ltd.

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

Table of contents

Linking static uFCoder library for iOS app development **3**

iOS uFCoder library usage **4**

Necessary information for projects "info.plist" file **4**

BLE usage 4

NFC usage 4

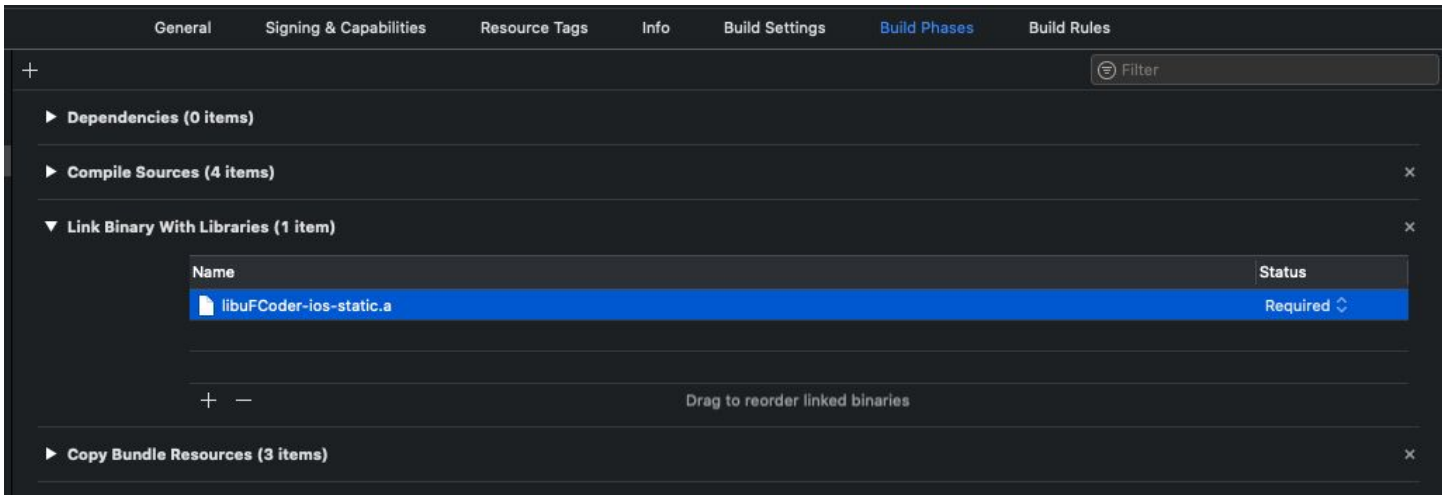
Revision history **7**

Digital Logic Ltd.

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

Linking static uFCoder library for iOS app development

1. Open your project settings in XCode, and go to "**Build Phases**"
2. Under "Link With Libraries" click on "**+**" to add the "**uFCoder-ios-static.a**" file in your project. (If you are using our ufr-lib repository, path of this file should be something like "/ufr-lib/ios/uFCoder-ios-static.a")



3. Depending on whether using **Swift** or **Objective-C**, you will need to include "uFCoder.h" header file in your project

3.1. Swift

3.1.1. Add a bridging header to Xcode project (**File** > New > File > **Header file)**
3.1.2. For example name the new file "YourProjectName-Bridging-Header.h"
3.1.3. Create the file and add "#import "<path_to_uFCoder.h_file>"
(e.g "#include "ufr-lib/include/uFCoder.h")
3.1.4. Navigate to your project **build settings** and find the "**Swift** Compiler - General" section
3.1.5. Add path of our newly created bridging header file in the value field of "Objective-C Bridging Header" field
3.1.6. Now you are free to use uFCoder functions in your iOS application.

3.2. Objective-C

3.2.1. No bridging header is necessary for Objective-C.
Simply add "#import "<path_to_uFCoder.h_file>" after you link your project with uFCoder library, as explained in step 2 and you're finished with setting up usage of uFCoder library in your application.

Note: Using uFCoder library is not possible when working with iOS simulators.
It was built specifically for iOS devices.. (building for option "Generic iOS device" is possible)

Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

iOS uFCoder library usage

uFCoder library for iOS currently only supports our uFR Nano Online NFC reader, more specifically, it supports UDP, TCP and BLE communication with this device. There is also support for using NFC antenna on your iOS device using our library for sending and receiving data, however it only supports APDU commands using the ApduPlainTransceive() function (demo code will be provided in the following text). When using BLE or phone NFC, your project's "info.plist" file must be updated accordingly, with necessary information about permissions and capabilities of your app.

Necessary information for projects "info.plist" file

BLE usage

If you plan on using BLE, you need to add the following in your "info.plist" file:

- "privacy - bluetooth always usage description" - message that tells the user why the app needs access to Bluetooth.
- "privacy - bluetooth peripheral usage description" - message that tells the user why the app is requesting the ability to connect to Bluetooth peripherals.

To connect to uFR Nano Online when the reader is in BLE mode, use **ReaderOpenEx()** function with following parameters:

- Reader type: 0
- Port name: ONxxxxxx - serial number of the reader (e.g ON123456_BLE)
- Port interface: 'L' or 76 (decimal)
- Additional argument: null (not a necessary parameter)

Function will return UFR_OK on success and readers RGB colors will be steady light-blue.

NFC usage

For now, you can use NFC only to send APDU commands via the device's NFC antenna.
You will need to add following in the "info.plist" file:

- "com.apple.developer.nfc.readersession.formats" - NFC data formats an app can read. This entitlement requires you to add "Near Field Communication Tag Reading" capability. This

Digital Logic Ltd.
Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

entitlement should be an array of strings in the info.plist file, add the "TAG" string as a value for this entitlement.

- "ISO7816 application identifiers for NFC Tag Reader Session" - list of application identifiers that the app supports. This entitlement is an array, and should contain following values:
    - A0000002471001
    - D2760000850101
    - 00000000000000
- Privacy - NFC Scan Usage Description - message that tells the user why the app is requesting access to the device's NFC hardware.

Use ReaderOpenEx() function with the following parameters:

- Port name: 5
- Port name: ""
- Port Interface: 0
- Additional argument: null or 0

To send an APDU command, connecting to the tag is necessary, to achieve this - SetISO14443_4_Mode() function is used.

On successful connect, UFR_OK is returned, and the tag is ready to receive APDU command(s).

To send an APDU command - ApduPlainTransceive() function is necessary.
Once you're done with sending the APDU commands, call s_block_deselect() function.

Whole process should look like this: (Swift code bellow):

Digital Logic Ltd.

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

```
    @IBAction func onSendApdu(_ sender: Any) {
        var status: UFR_STATUS = ReaderOpenEx(5, "", 0, nil)
        print("ReaderOpenEx_status: \(status)")

        status = SetISO14443_4_Mode()
        print("SetISO14443_4_Mode: \(status)")
        if (status != UFR_OK)
        {
            print("SetISO14443_4_Mode failed")
            return
        }
            let str_apdu = "00A4040C07A0000002471001"
            let capdu = str_apdu.hexa
            let clen: UInt32 = UInt32(capdu.count)
            var rapdu = [UInt8](repeating: 0x00, count: 10)
            var rlen: UInt32 = 1

            status = APDUPlainTransceive(capdu, clen, &rapdu, &rlen)
            print("APDUPlainTransceive: \(status)")

            s_block_deselect(100)
    }
}

extension Sequence where Element == UInt8 {
    var data: Data { .init(self) }
    var hexa: String { map { .init(format: "%02x", $0) }.joined() }
}
```


Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com

Revision history

Date	Version	Comment
2019-04-09	1.0	Base document

Digital Logic Ltd.

Address: Nemanjina 57a, 12000 Pozarevac, Serbia · Tel: +38112541022· VAT: 111385444 Reg. 21473642
e-mail: office@d-logic.com · www.d-logic.com